Understanding GPU Energy Dynamics in HPC Applications

Ethan Puyaubreau, Université Paris-Saclay, France

Daniel Arndt & Jakob Bludau & Damien Lebrun-Grandié, ORNL, Oak Ridge

Oak Ridge National Laboratory - Computational Science and Engineering Division

Introduction

Growing demands for computational power in high-performance computing (HPC) shift the focus from raw performance to sustainable operation and energy-efficiency[1]. Current profiling methods struggle to link power usage with fine-grained computational events, as hardware counters have coarse sampling and software instrumentation adds significant overhead[2], [3].

Nevertheless, for larger code regions like full algorithm implementations total energy consumption can be reliably measured. This allows for direct comparisons of different approaches in terms of energy efficiency.

Using Kokkos Tools for Energy Profiling

Kokkos is a performance-portability library that allows applications to target various backends (CPUs, Nvidia GPUs, AMD GPUs, Intel GPUs) with a common interface that doesn't require changes in the application's source code. The Kokkos Tools framework offers flexible profiling for Kokkos applications[4]. Its modular design allows code execution at the entry and exit of user-defined regions, requiring no application changes.

By continuously collecting background power readings, we can analyze how GPU kernel launches affect power consumption. This lets users quantify power and energy for specific code regions, comparing algorithm efficiency to assess energy consumption.

We use the Kokkos Profiling API with a Variorum[5]/NVML[6] (NVIDIA Management Library) profiling tool to capture device power measurements. Aligning region timestamps with power data in a post-processing step provides actionable insights into GPU power behavior in HPC.

Problem of Measuring Power Directly

Figure 1 shows exemplary GPU power readings. The GPU power management unit can be queried only every 100ms for instantaneous power [6]. However, the true power consumption can vary between these sampling points. Furthermore, the reported power is given for the last 25ms of the 100ms interval [3]. To account for this, the power is sampled in 64 consecutive runs with `(repetition*5ms)` sleep in between [3]. Then the maximum over all reported power measurements is taken.

Because most GPU kernels in HPC applications execute in under 10 ms, their individual power profiles cannot be captured directly. Nonetheless, measuring larger, user-defined regions lets users compare algorithms by energy use. Figure 2 highlights the different power levels the GPU operates at for continuous workloads with different compute and memory loads.

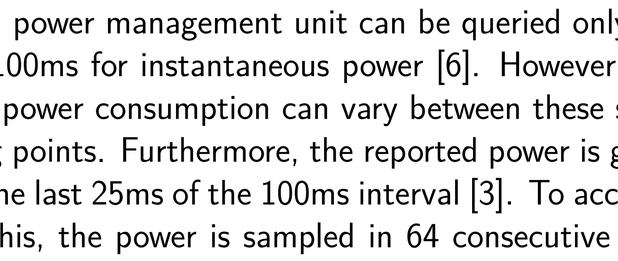

Figure 1. Sampling GPU power for the last 25ms of every 100ms misses power spikes between samples.

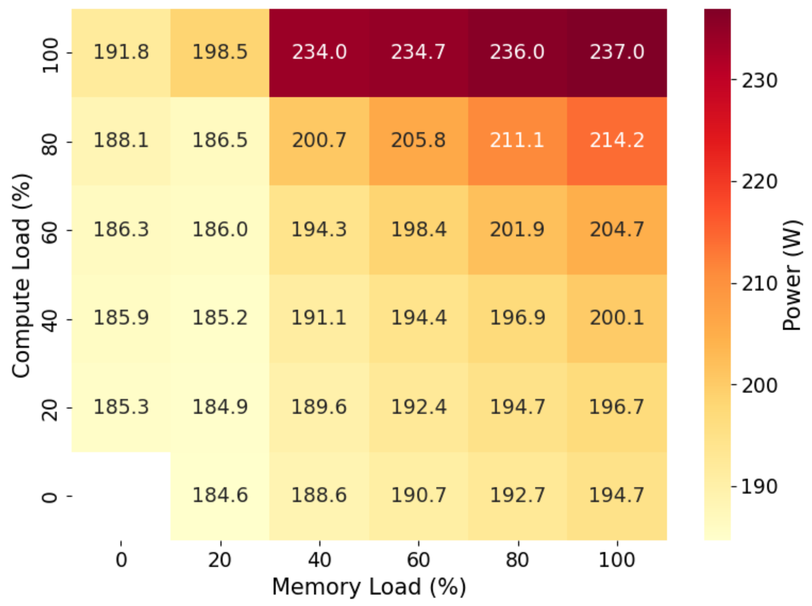

Figure 2. Heatmap of average power level (W) for combinations of compute-bound vs memory-bound workloads on an NVIDIA H100 NVL GPU. Computed with the bytes and flops benchmark of Kokkos [7].

Minimal runtime does not imply energy efficiency

GPU power dynamics mandate per-algorithm, per-hardware measurements, yet tools are lacking the required resolution

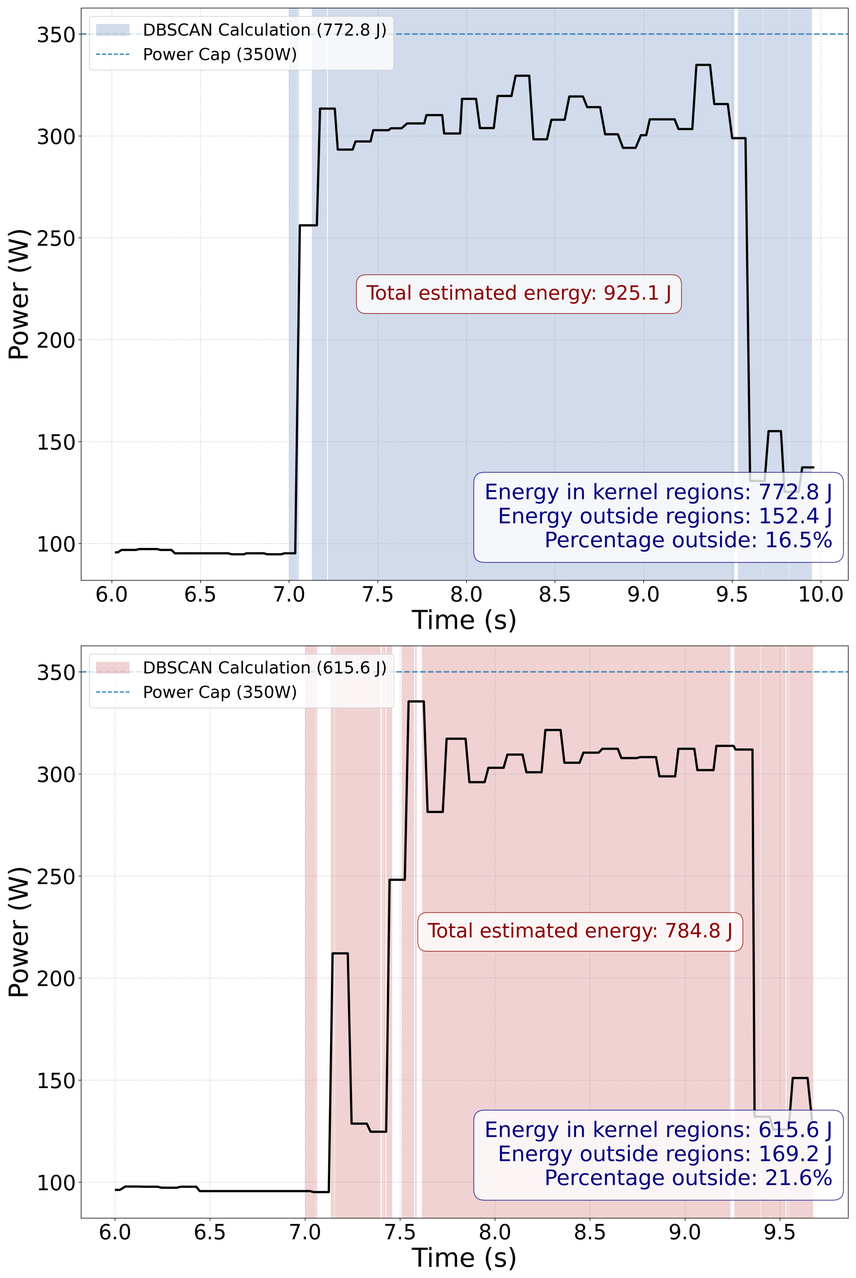

Figure 3. GPU Power level over time during an ArborX DBSCAN benchmark using the `fdbscan` and the `fdbscan-dense` implementation (top to bottom) as described in [8]. Black line shows power level, colored regions indicate different Kokkos kernels. The energy is numerically integrated from the power and duration. Both algorithms have the same runtime and give the same result to the user.

Power Dynamic of HPC GPUs

Using NVML[6], we sample the instantaneous GPU power every 100 ms, thus enabling accurate tracking of steady-state consumption under different memory vs. compute loads:

- Steady-state power varies with workload type; Figure 2 shows a heatmap of average power level for different compute and memory loads on an NVIDIA H100 NVL GPU.
- Memory-intensive algorithms (storing intermediates) can drive higher power, while compute-intensive (recomputing intermediates) approaches may be more energy efficient. Figure 3 compares an exemplary case.
- These findings are hardware and configuration-dependent and must be validated for each configuration/algorithm/hardware.

Conclusion and Future Work

Our investigation confirms that current software-based profiling tools like NVML have inherent limitations in temporal resolution [3]. This prevents per-kernel analysis for kernel runtimes <25ms [3]. Nevertheless, continuously measuring power in the background while recording the start and end of user-defined regions lets users compare algorithms with respect to their energy efficiency. After characterizing the power dynamics of a respective GPU (as done for memory and compute in Figure 2 for an H100 NVL GPU), users know which hardware operations drive the power consumption. Using this information different algorithms can be selected and compared using the developed tool. Kokkos users can leverage our tool without code changes, while non-Kokkos codes only need to annotate their regions of interest.

Acknowledgments

This material is based upon work supported by the U.S. Department of Energy, Office of Science, Office of Advanced Scientific Computing Research (ASCR) as part of the Next Generation of Scientific Software Technologies program, Stewardship of Programming Systems and Tools (S4PST) project.

This research used resources on the Frank cluster at the University of Oregon.

References

[1] C. Porter, *Energy Efficiency in HPC*, NVIDIA Developer Blog, Nov. 14, 2023. [Online]. Available: https://developer.nvidia.com/blog/energy-efficiency-in-high-performance-computing-balancing-speed-and-sustainability/.

[2] L. Mukhanov, D. S. Nikolopoulos, and B. R. de Supinski, "ALEA: Basic-Block Energy Profiling," *arXiv*, 2015. [Online]. Available: https://arxiv.org/abs/1504.00825.

[3] Z. Yang, K. Adamek, and W. Armour, "Part-time power measurements: Nvidia-smi's lack of attention,", 2023. DOI: 10.48550/ARXIV.2312.02741. [Online]. Available: https://arxiv.org/abs/2312.02741.

[4] *Kokkos Tools*, https://github.com/kokkos/kokkos-tools, 2025.

[5] *Variorum*, https://github.com/LLNL/variorum.

[6] *NVIDIA NVML API*, https://docs.nvidia.com/deploy/nvml-api/.

[7] C. R. Trott, D. Lebrun-Grandié, D. Arndt, J. Ciesko, V. Dang, N. Ellingwood, R. Gayatri, E. Harvey, D. S. Hollman, D. Ibanez, N. Liber, J. Madsen, J. Miles, D. Poliakoff, A. Powell, S. Rajamanickam, M. Simberg, D. Sunderland, B. Turcksin, and J. Wilke, "Kokkos 3: Programming model extensions for the exascale era," *IEEE Transactions on Parallel and Distributed Systems*, vol. 33, no. 4, pp. 805–817, 2022. DOI: 10.1109/TPDS.2021.3097283.

[8] A. Prokopenko, D. Lebrun-Grandie, and D. Arndt, "Fast tree-based algorithms for dbscan for low-dimensional data on gpus," in *Proceedings of the 52nd International Conference on Parallel Processing*, ser. ICPP 2023, ACM, Aug. 2023, pp. 503–512. DOI: 10.1145/3605573.3605594. [Online]. Available: http://dx.doi.org/10.1145/3605573.3605594.

[9] V. Singhania, S. Aga, and M. A. Ibrahim, "FinGraV: Fine-Grain GPU Power Visibility," *arXiv*, 2024. [Online]. Available: https://arxiv.org/abs/2412.12426.

OAK RIDGE National Laboratory

kokkos

U.S. DEPARTMENT of ENERGY